Cryptography Intro

CS642: Computer Security

Spring 2019

AWS Urges Devs To Scrub Secret Keys From GitHub

28

timothy (36799) posted 2 hours ago | from the key-is-under-the-mat dept.

An anonymous reader writes "GitHub contains thousands of 'secret keys', which are stored in plain text and can be used by miscreants to access AWS accounts and either run up huge bills or even delete/damage the users files. Amazon is urging users of the coding community site to clean up their act."

Cryptography

Basic goals and setting

TLS (HTTPS)

Provable security

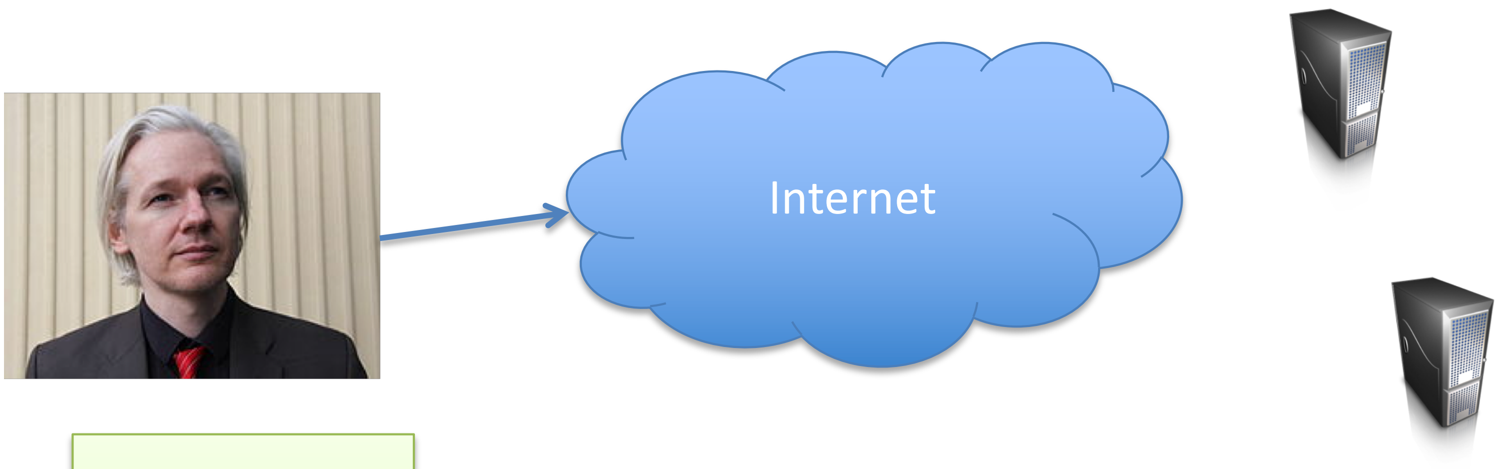
One time pad

Block ciphers

Cryptography: “Hidden writing”

- Study and practice of building security protocols that resist adversarial behavior
- Blend of mathematics, engineering, computer science

Cryptography Example



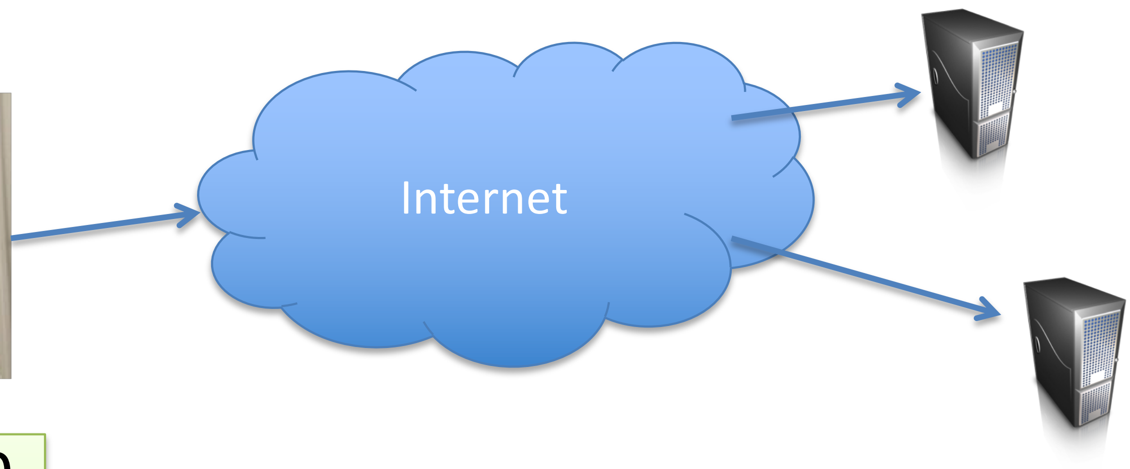
US diplomatic cables

Don’t want to reveal data early

Want to store it in way that it can quickly be revealed later

Cryptography Example

01101010
10101010
10101010
11111101



01101010
10101010
10101010
11111101

Don’t want to reveal data early

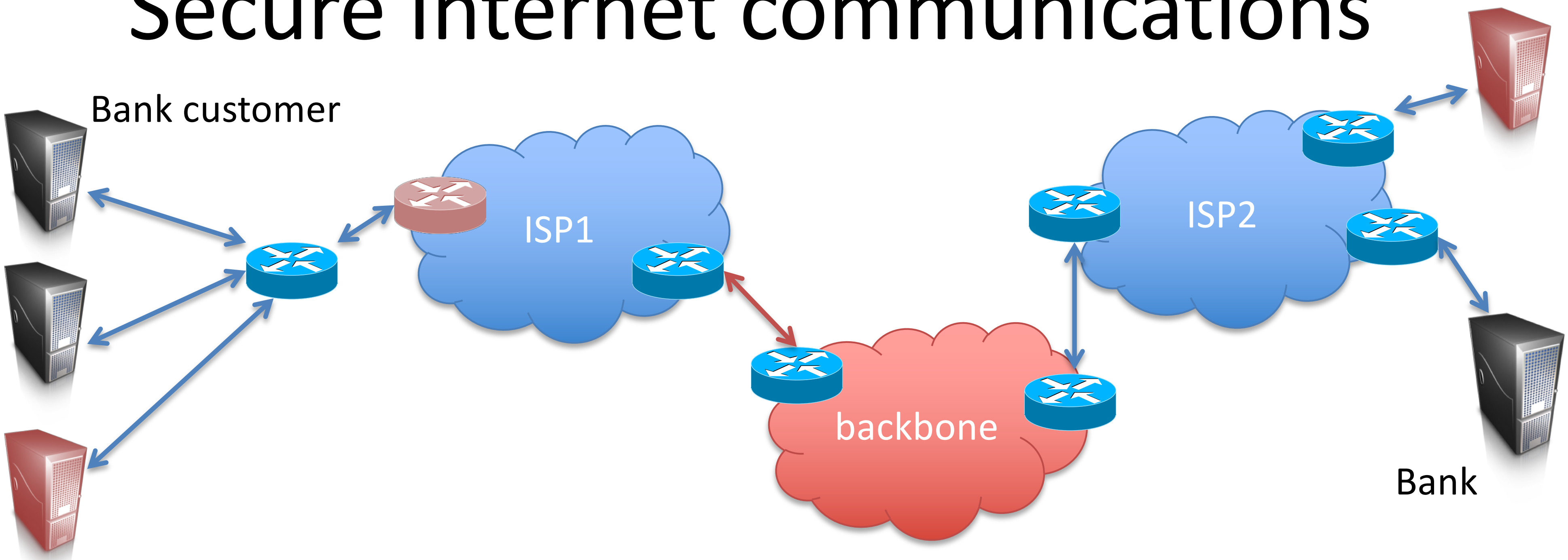
Want to store it in way that it can quickly be revealed later

01101010
10101010
10101010
11111101

Modern cryptography enables this:

- Encrypt file
- Store key in secure place

Crypto Example 2: Secure Internet communications



Customer and bank want to communicate securely:

- **Confidentiality** (messages are private)
- **Integrity** (accepted messages are as sent)
- **Authenticity** (is it the bank? is this the customer?)
- Sometimes: anonymity (hide identities)
- Sometimes: steganography (hide that communication took place)

TLS, SSH, IPsec, PGP

Encrypted hard drives

Corporate intellectual property
Customer financial records
Personal notes

Encrypt hard drives or individual files

- Confidentiality
- Even if attacker has physical access to device

Bitlocker, TrueCrypt, OSX, iOS, Seagate

example 3

Crypto

- Powerful tool for confidentiality, authenticity, and more
- But:
  - must design securely
  - must implement designs securely
  - must use properly (e.g., key management)

Auguste Kerckhoffs' (Second) Principle

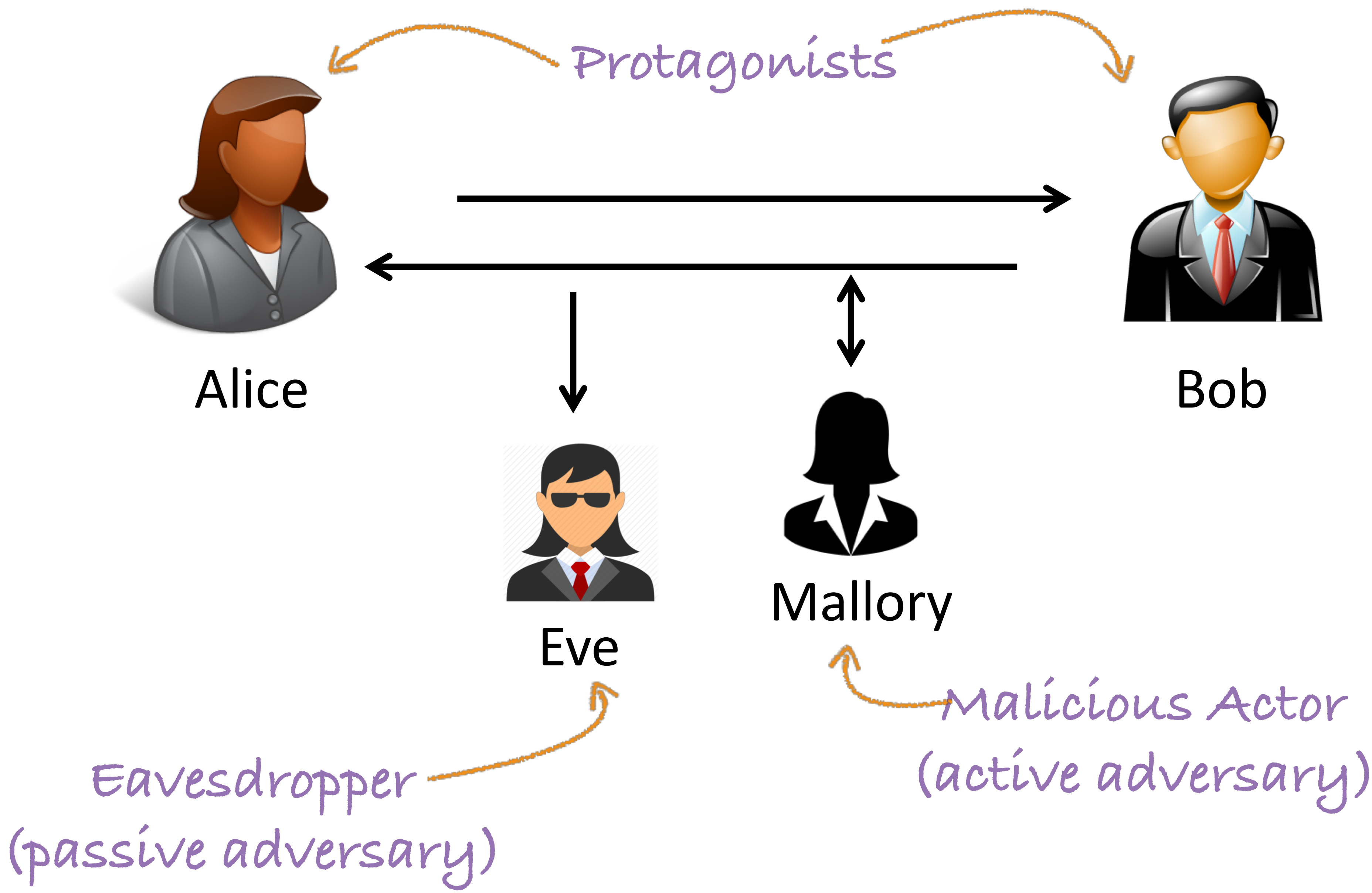
"The system must not require secrecy and can be stolen by the enemy without causing trouble"

A cryptosystem should be secure even if its algorithms, implementations, configuration, etc. is made public --- the only secret should be a key

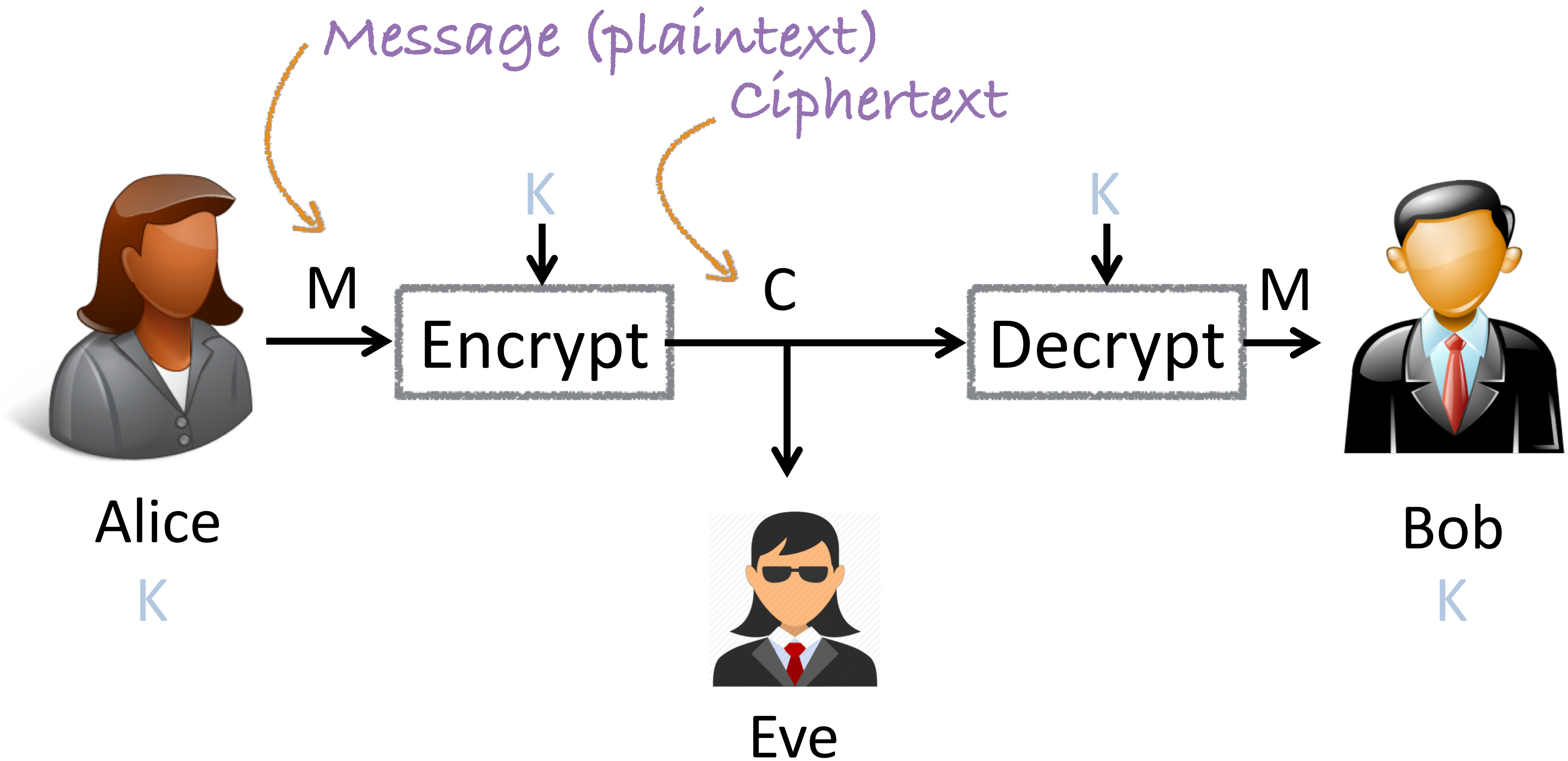
Why?

primitives

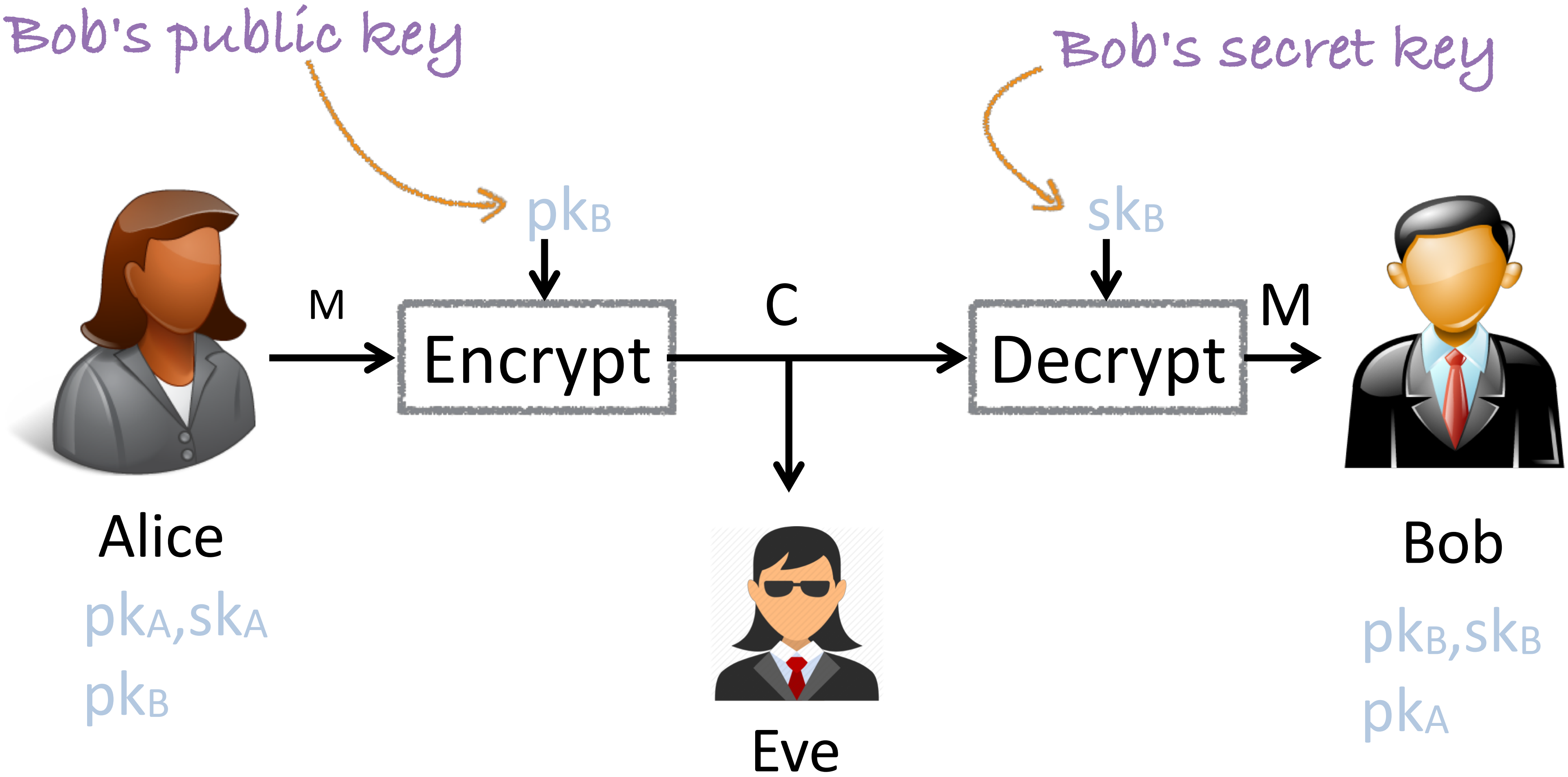
- Encryption
  –confidentiality
  –symmetric + asymmetric versions
- Message authentication codes
  –integrity, authentication
  –symmetric
- Digital signatures
  –integrity, authentication
  –asymmetric
- Key exchange



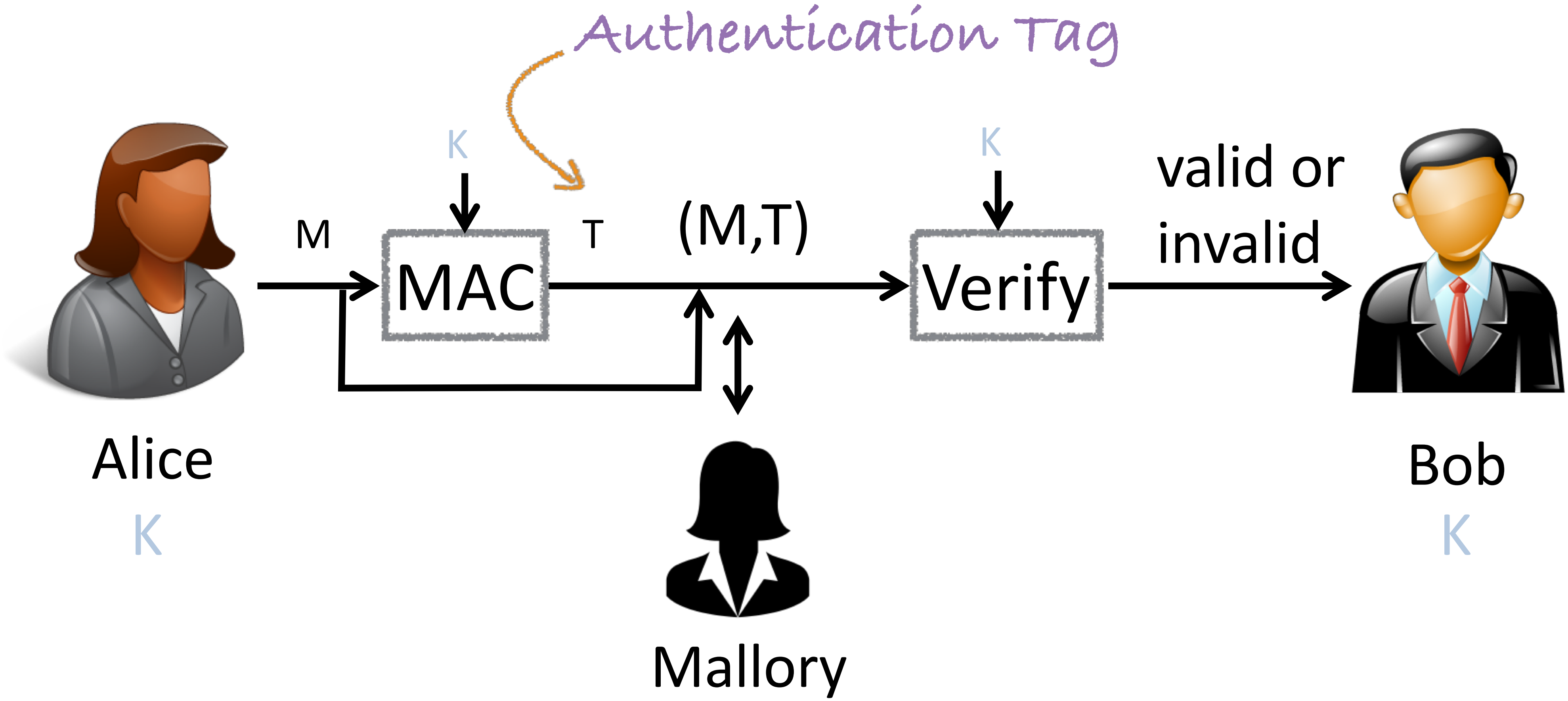
conventions



symmetric encryption

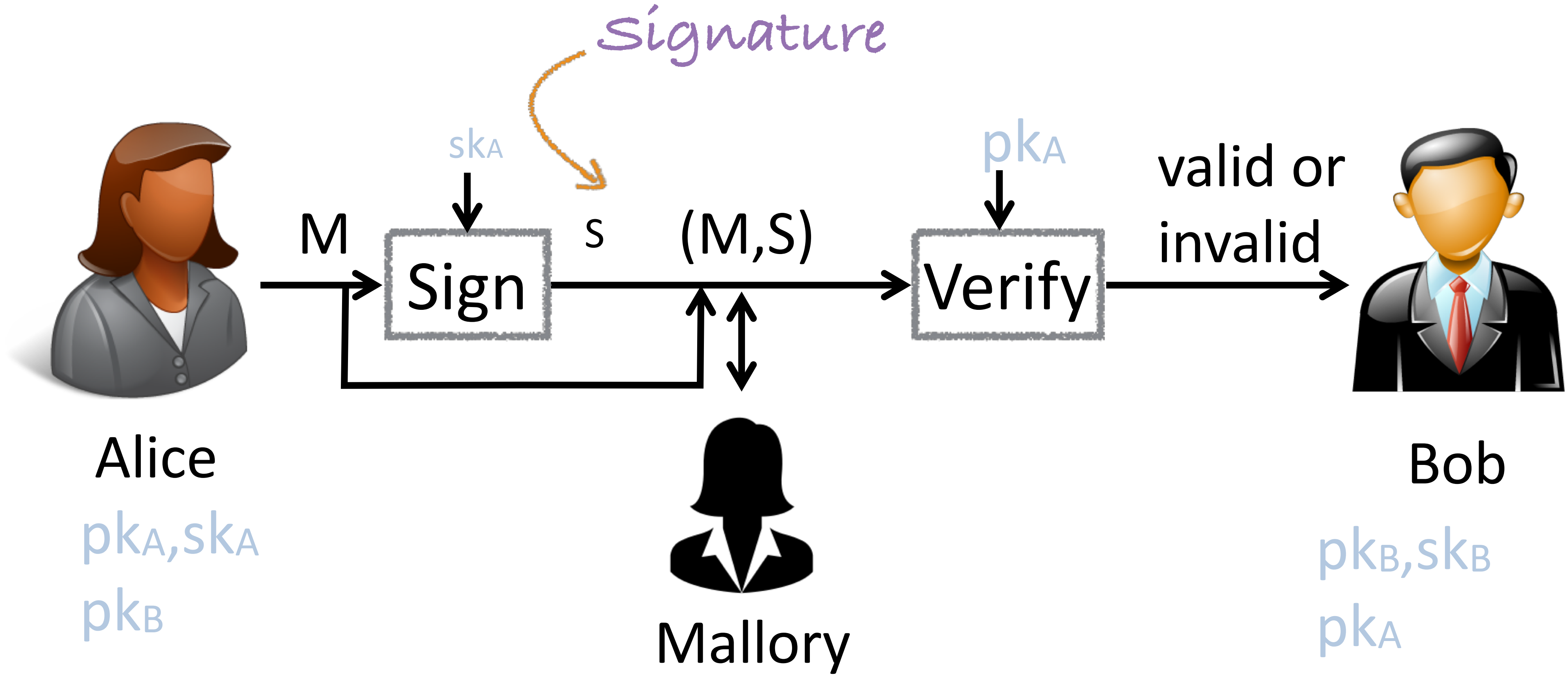


asymmetric encryption



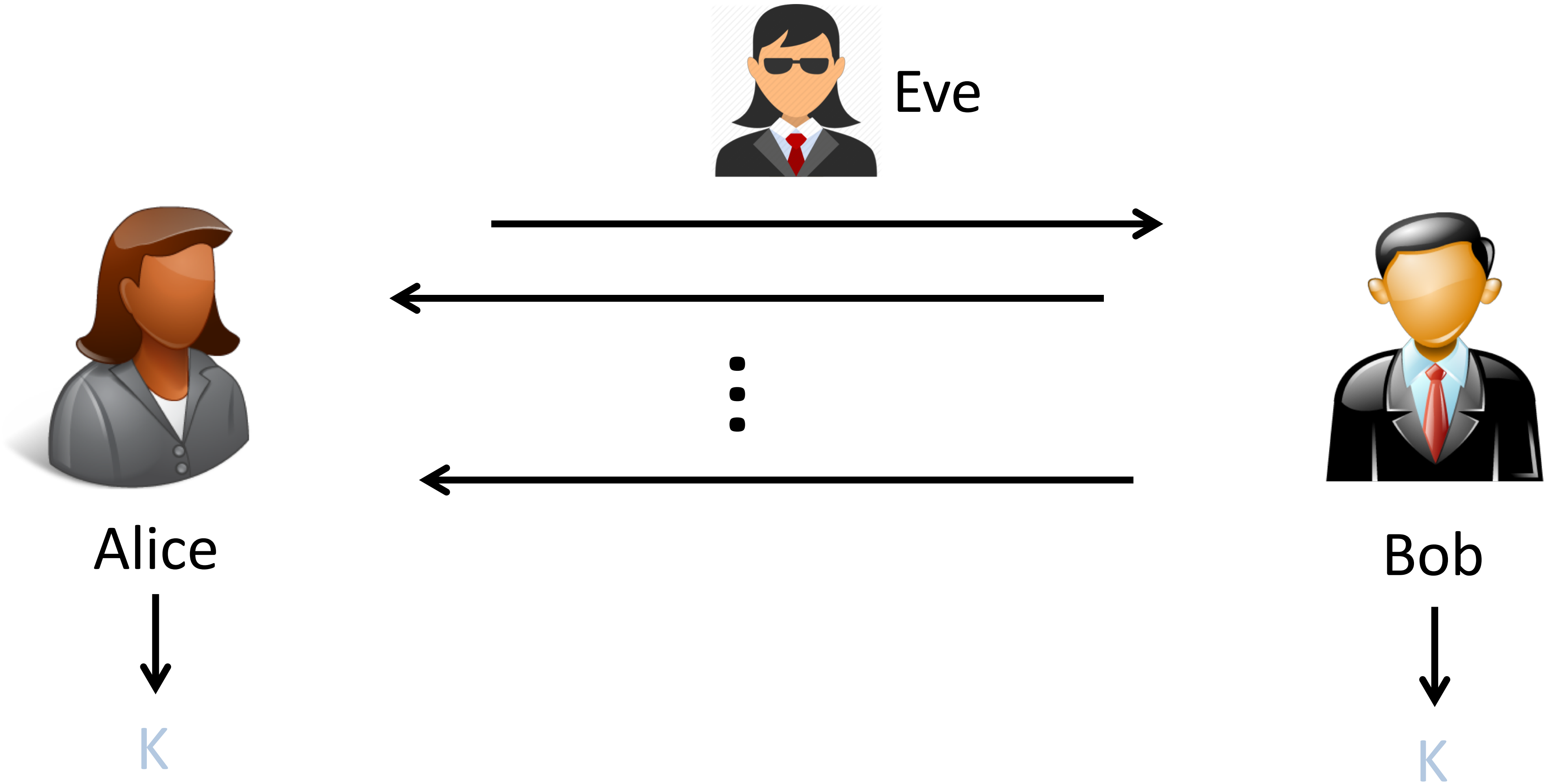
Message Authentication Code (MAC)
message integrity & authenticity / symmetric

mac



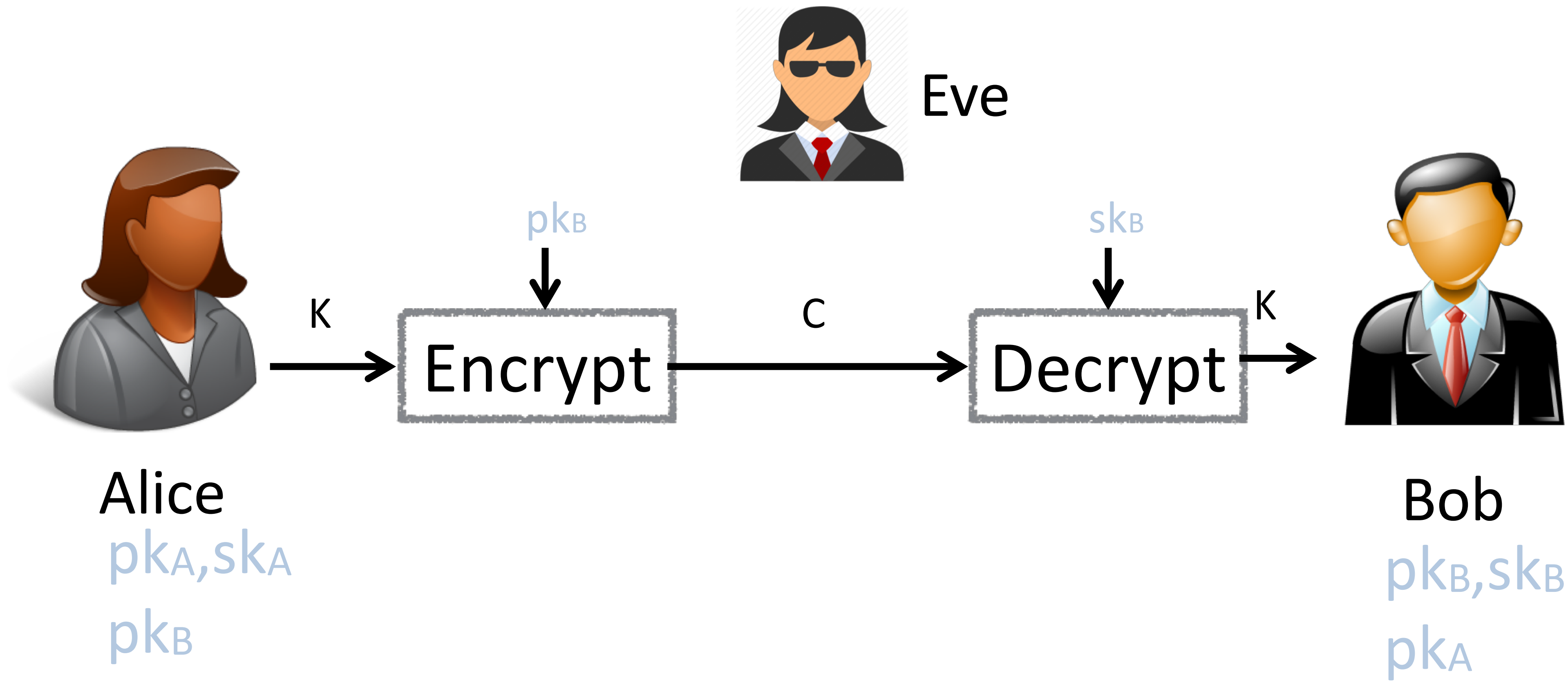
message integrity & authenticity / asymmetric

digital signatures



Alice and Bob exchange messages in the presence of an eavesdropper, and (magically) both generate an identical secret (symmetric) key that Eve cannot know

key exchange



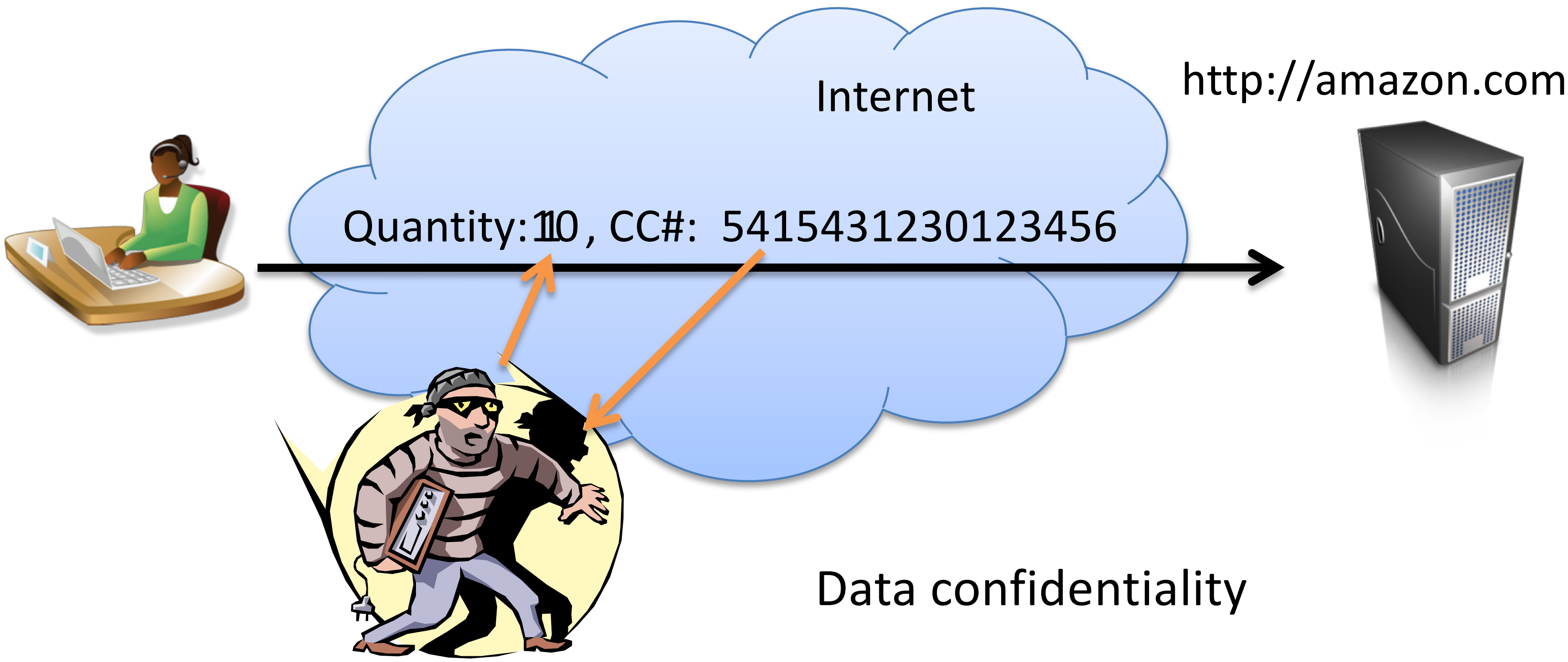
K := rand()

Two main techniques for key exchange

1. Public key transport (shown here)
2. Diffie-Hellman key agreement

key transport

An example: Online shopping

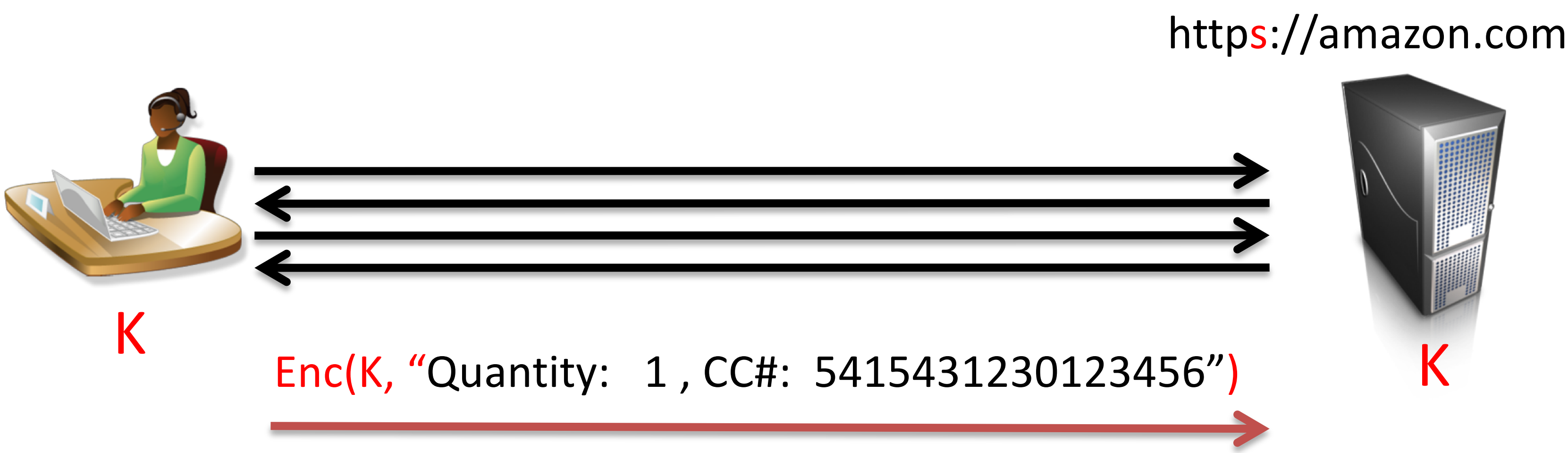


Data confidentiality

Data integrity

We need secure channels for transmitting data

An example: On-line shopping with TLS



Step 1:
Key exchange protocol to share secret K

Step 2:
Send data via secure channel

TLS uses many cryptographic primitives:

> **key exchange:** hash functions, digital signatures, public key encryption
> **secure channel:** symmetric encryption, message authentication

Mechanisms to resist replay attacks, man-in-the-middle attacks, truncation attacks, etc…

A short history of TLS up to 2009

Event	Version	Year
SSL ver 2.0 designed by **Hickman** at Netscape	SSL ver 2	1994
Wagner, Goldberg break SSL ver 2		1995
Freier, Karlton, Kocher design SSL ver 3.0	SSL ver 3	
Bleichenbacher breaks RSA PKCS #1 encryption, used in SSL ver 3		1998
TLS ver 1 released as IETF standard, based on SSL 3, **many cryptographers** involved	TLS ver 1.0	1999
Vaudenay, Klima et al. padding attacks		2001
Rogaway IV re-use insecurity		2002
Brumley, Boneh remote timing attacks		2003
TLS ver 1.1 released as standard	TLS ver 1.1	2006

How many cryptographers involved?

⋮

(more attacks and fixes)

Bank customer

TLS handshake for RSA transport

Bank

Pick random Nc

ClientHello, MaxVer, Nc, Ciphers/CompMethods →

Pick random Ns

← ServerHello, Ver, Ns, SessionID, Cipher/CompMethod

Check CERT using CA public verification key

← CERT = (pk of bank, signature over it)

Pick random PMS
C <- E(pk,PMS)

C →

PMS <- D(sk,C)

ChangeCipherSpec,
{ Finished, PRF(MS, "Client finished" || H(transcript)) } →

Bracket notation means contents encrypted

← ChangeCipherSpec,
{ Finished, PRF(MS, "Server finished" || H(transcript')) }

MS <- PRF(PMS, "master secret" || Nc || Ns)

Bank customer

TLS Record layer

Bank

MS <- PRF(PMS, “master secret” || Nc || Ns)

K1,K2 <- PRF(MS, “key expansion” || Ns || Nc)

C1 <- E(K1,Message)

C1

Message <- D(K1,C1)

C2 <- E(K2,Message’)

C2

Message’ <- D(K2,C2)

Primitives used by TLS

CERT = (pk of bank, signature over it)

Digital signatures

C

Public-key encryption (RSA)

ChangeCipherSpec,
{ Finished, PRF(MS, “Client finished” || H(transcript)) }

PRF
Hash function

C1

C2

Symmetric encryption

TLS was built via "design-break-redesign-break…"

We're now at TLS ver 1.2
~~No (publicly) known attacks~~

Did the TLS designers get it right?

In last few years host of attacks that affect TLS 1.2 as well have been discovered
[Paterson, Ristenpart, Shrimpton 2011]
Lucky 13 attack [AlFardan, Paterson 2013]
…

Even for "simple" applications (secure channels), secure cryptography is **really hard to design**. The problems are rarely in primitives.

Many other tools have similar story:

SSH, IPSec, Kerberos, WEP/WPA (WiFi security), GSM (cell phone networks), …

Provable security cryptography

Supplement "design-break-redesign-break…" with a more mathematical approach

1. Design a cryptographic scheme
2. Provide proof that no one is able to break it

Shannon 1949

<u>Formal definitions</u>

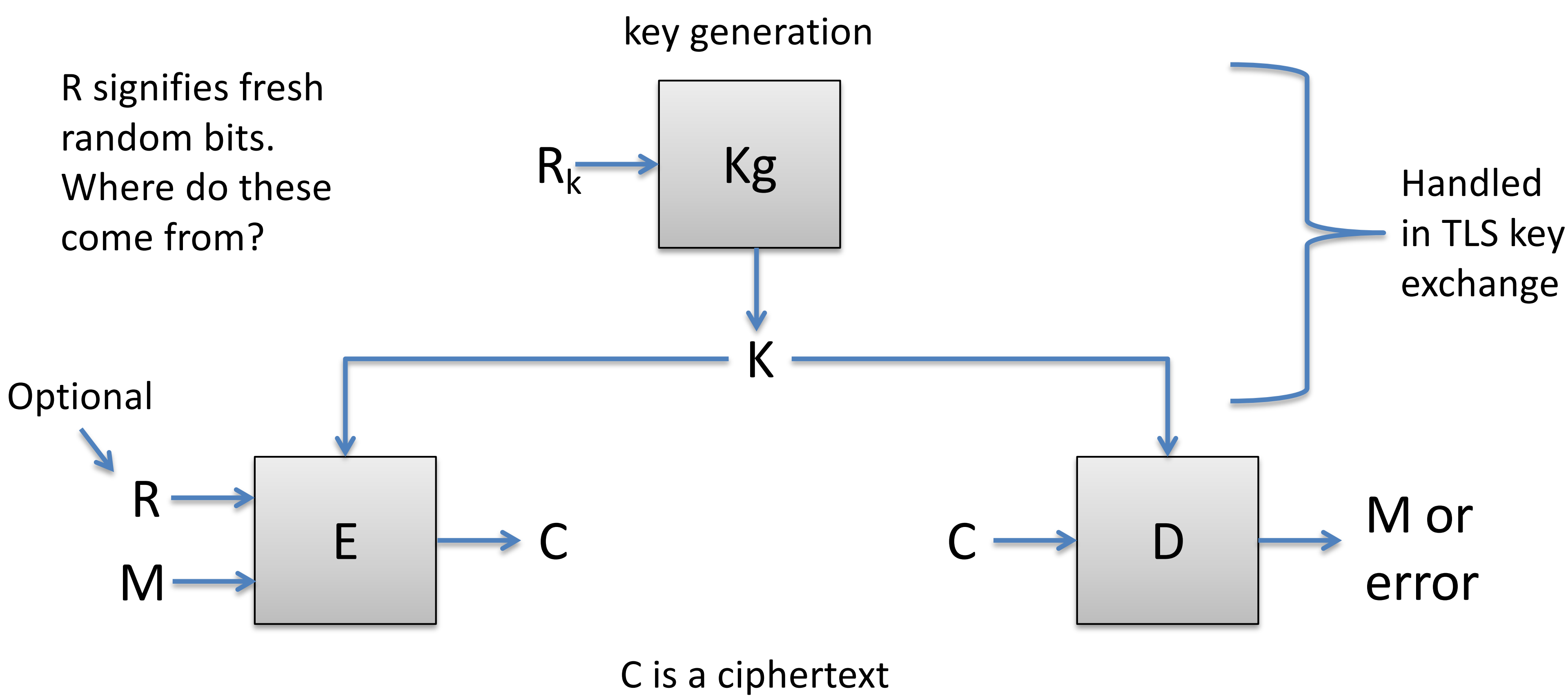
Scheme semantics

Security

<u>Security proofs</u>

Show it is mathematically impossible to break security

Symmetric encryption



Correctness: D(K , E(K,M,R)) = M with probability 1 over randomness used

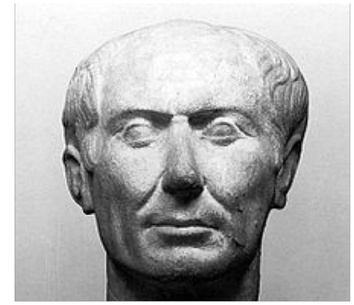
Kerckhoffs' principle: what parts are public and which are secret?

Some attack settings

- Unknown plaintext
  - attacker only sees ciphertexts
- Known plaintext
  - attacker knows some plaintext-ciphertext pairs
- Chosen plaintext
  - attacker can choose some plaintexts and receive encryptions of them

Substitution ciphers

Julius Caeser



Kg: output randomly chosen permutation of digits

	0	1	2	3	4	5	6	7	8	9	plaintext digit
K =	8	2	7	4	1	6	0	5	9	3	ciphertext digit

E(K, 2321-4232-1340-1410) = 7472-1747-2418-2128

Jane Doe	2414-2472-2742-7428
Michael Swift	3612-4260-2478-7243
John Jones	6020-7412-7412-2728
Eve Judas	7472-1747-2418-2128

1343-1321-1231-2310

Knowing one plaintext, ciphertext pair leaks key material!

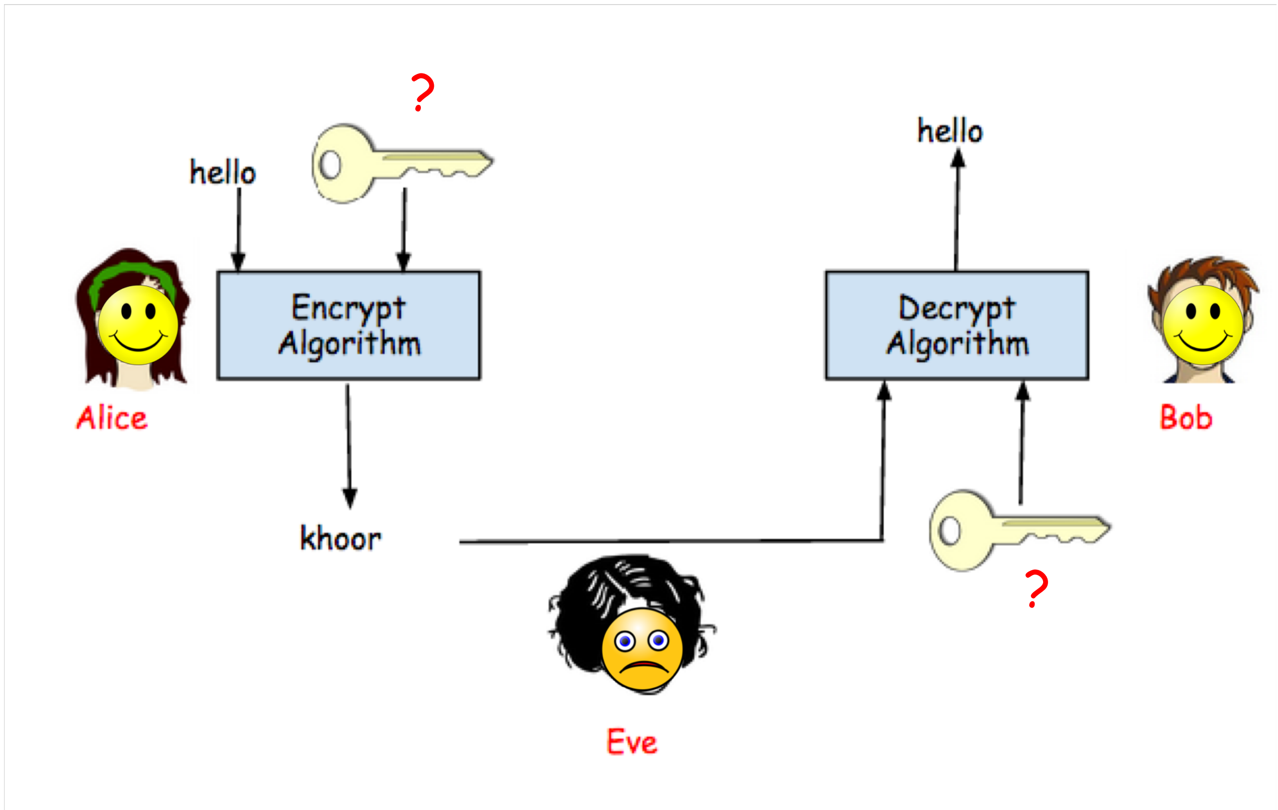
Attacker knows 2321-4232-1340-1410

7472-1747-2418-2128

0	1	2	3	4	5	6	7	8	9
?	?	?	?	?	?	?	?	?	?

Cracking Simple Substitution

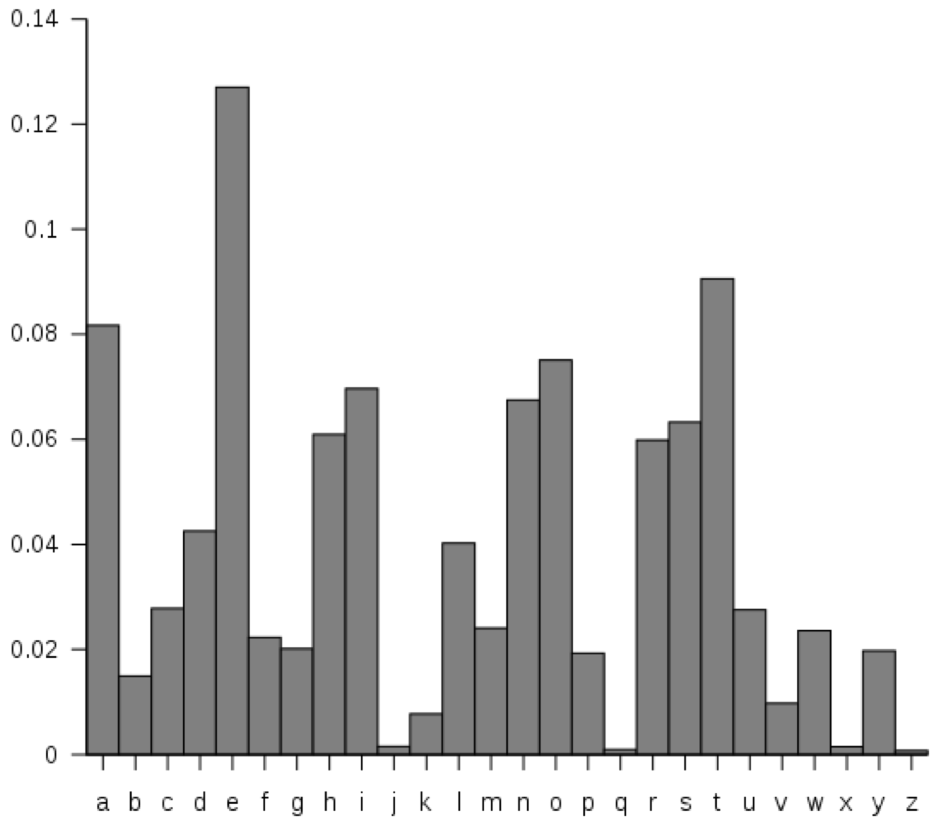
- *Brute force attack:* Eve would need 26! keys.
- That's 4.0329146e+26 keys. Too hard!



Cracking Simple Substitution

- *But, wait a minute...*

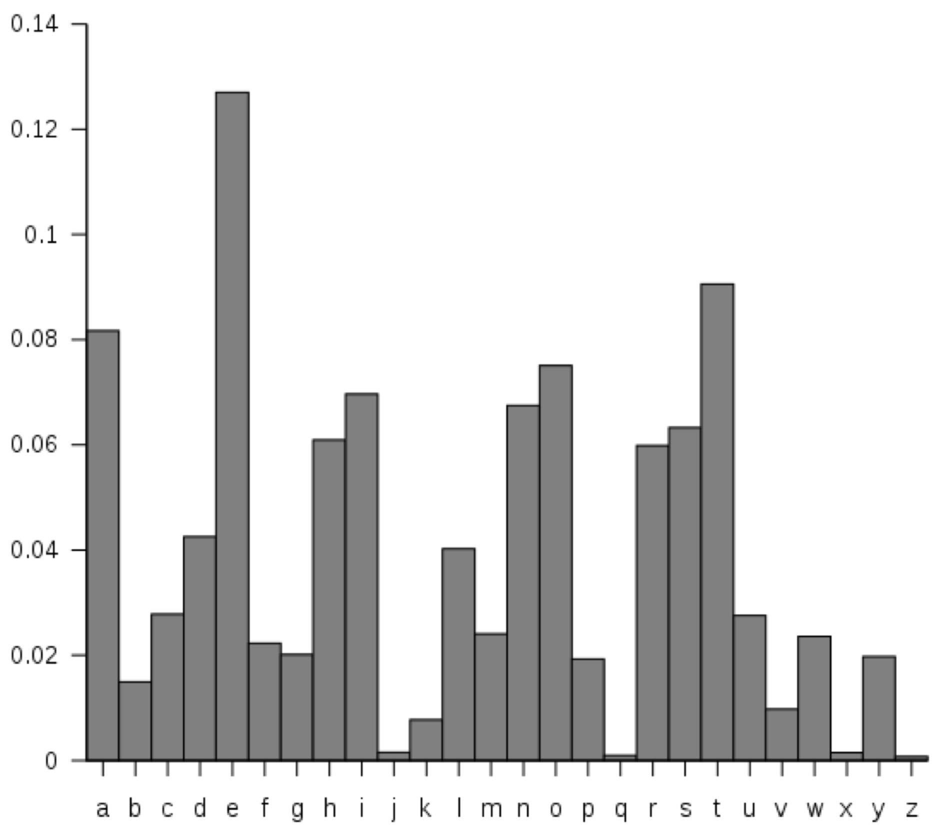
English plaintext
letter frequencies



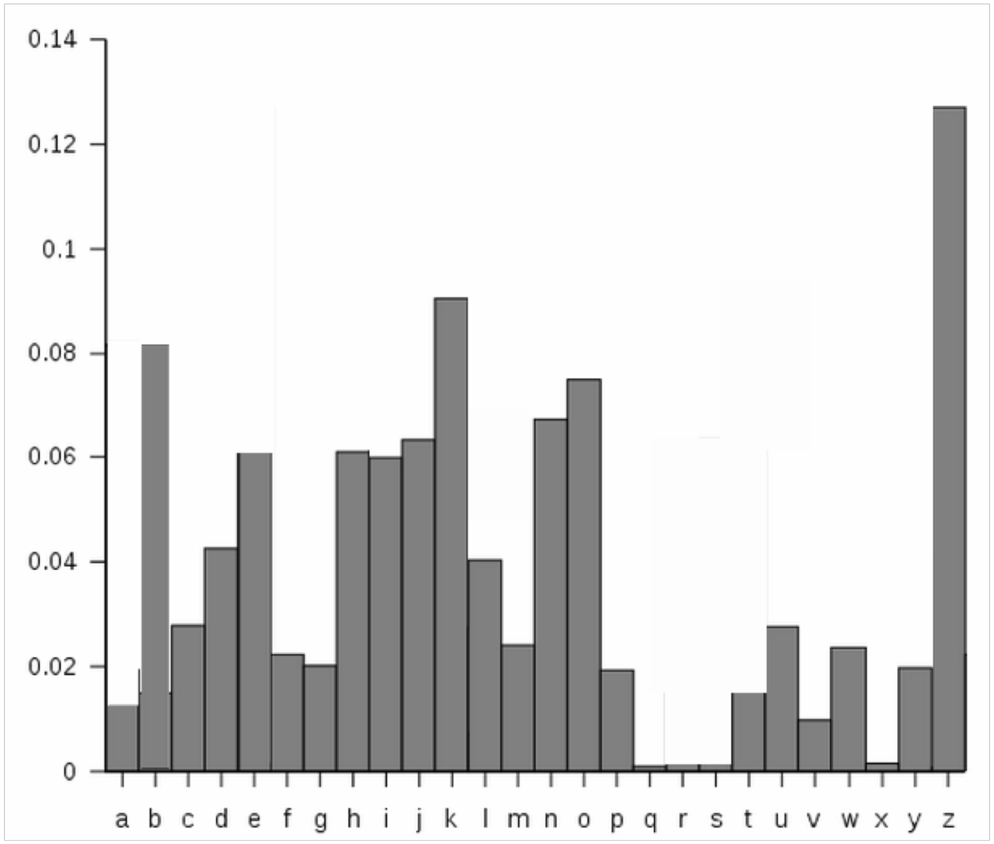
Cracking Simple Substitution

- *But, wait a minute...*

English plaintext
letter frequencies



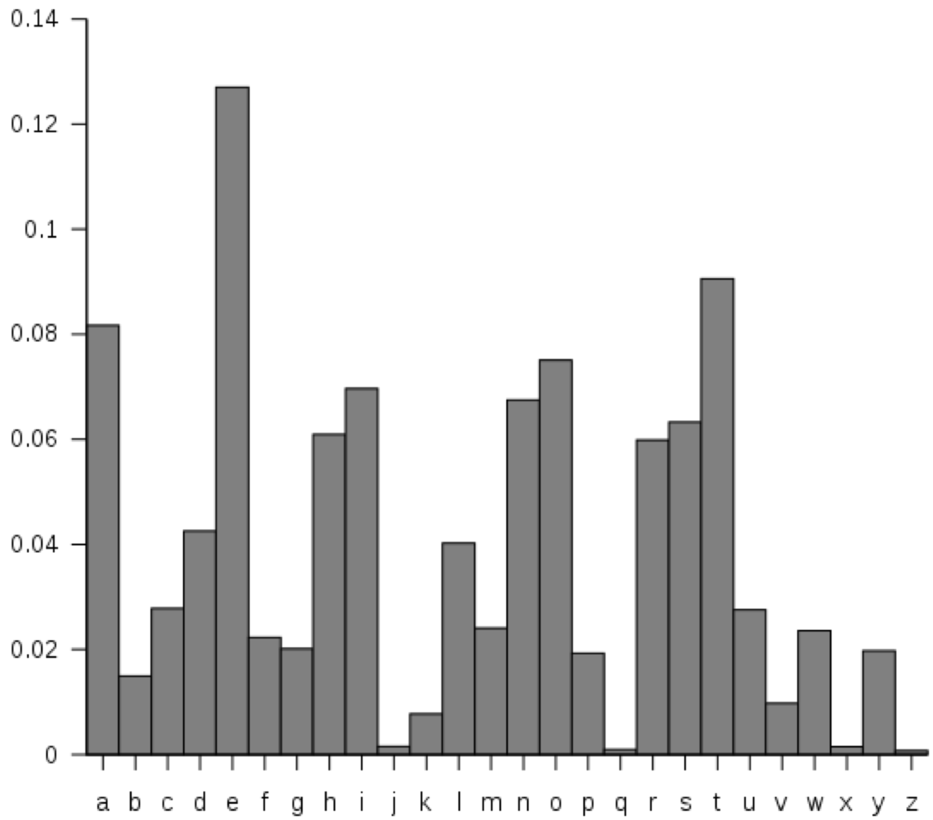
Ciphertext
letter frequencies



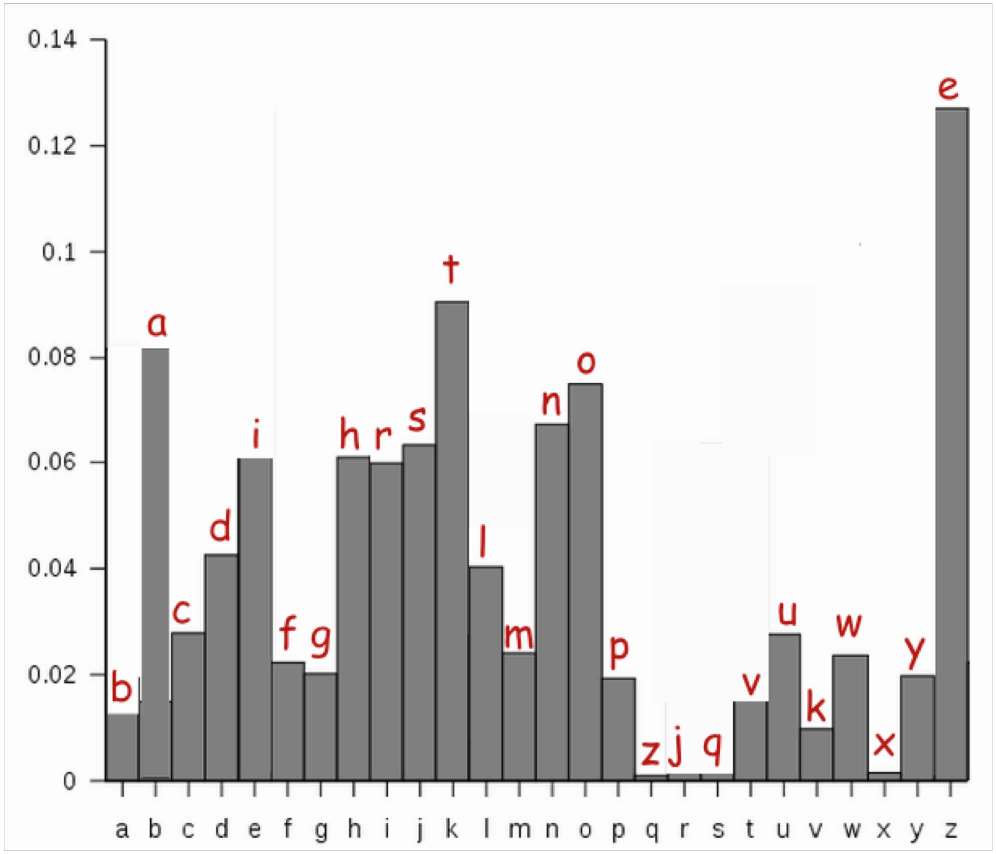
Cracking Simple Substitution

- *But, wait a minute… frequency analysis works!*

English plaintext letter frequencies

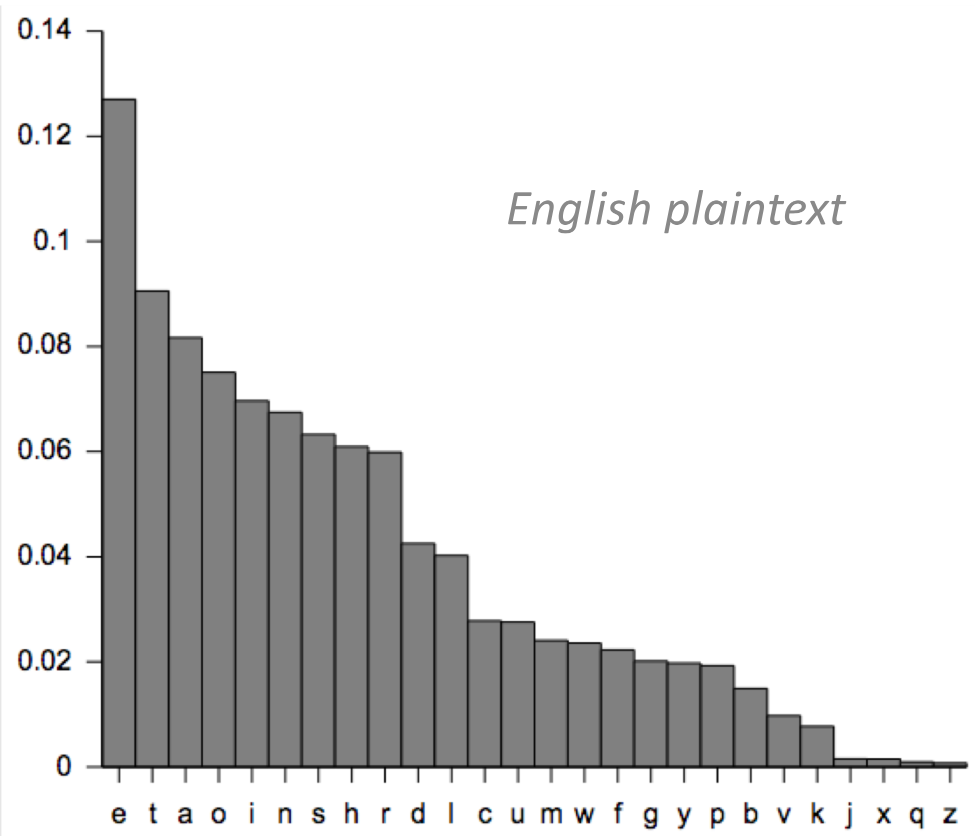


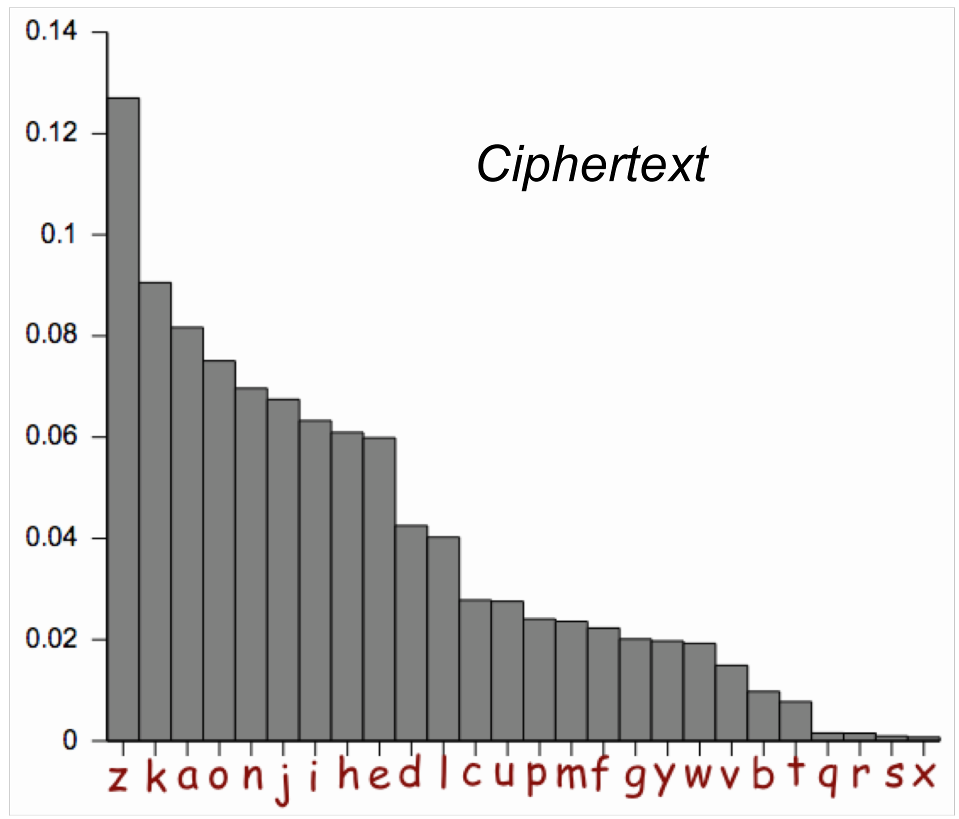
Ciphertext letter frequencies



Cracking Simple Substitution

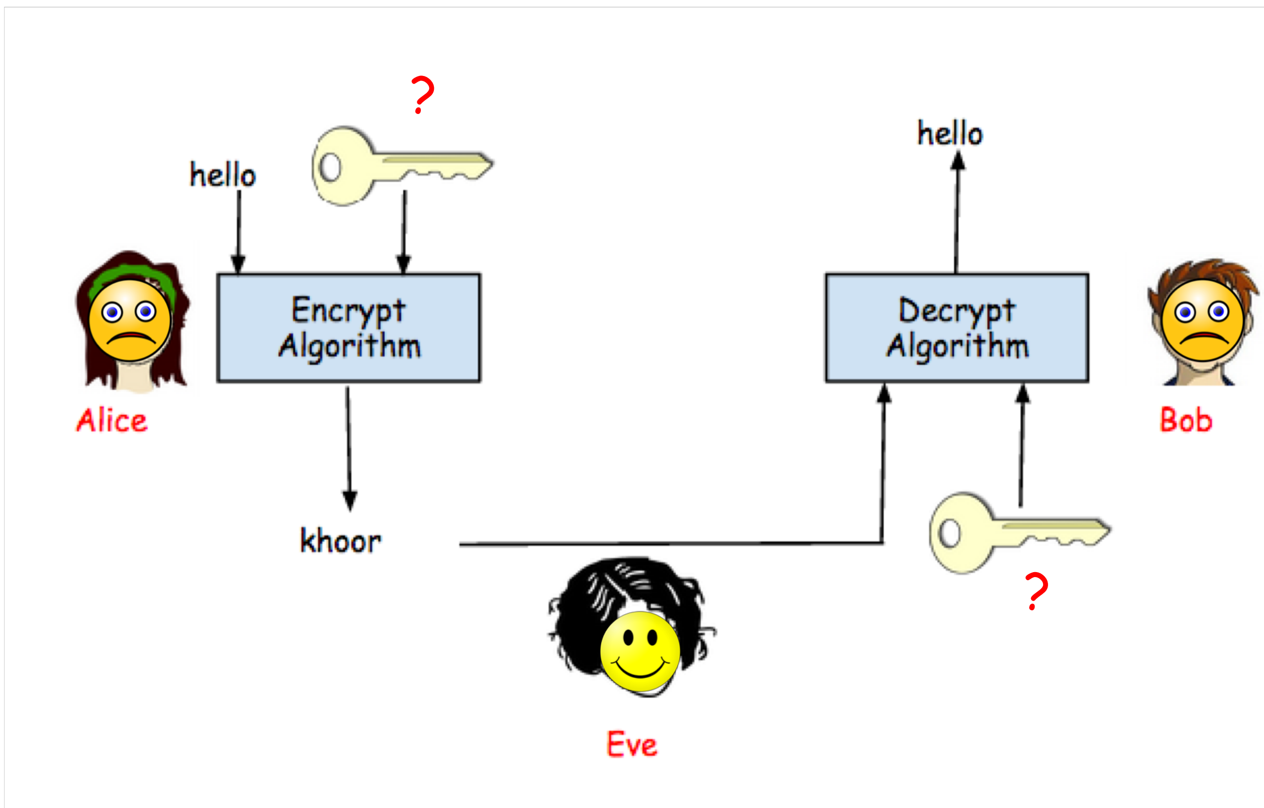
- *Can sort by frequencies*





Cracking Simple Substitution

- Eve wins … you don't need brute force
- *Frequency analysis* will break simple substitution



enigma

- Enigma is state of the art cryptography developed by the Germans
- Broken by the Allies
- Raises theoretical questions about cryptography

One-time pads

Fix some message length L

Kg: output random bit string K of length L

$E(K,M) = M \oplus K$ $\qquad$ $D(K,C) = C \oplus K$

Shannon's security notion

Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

$$\Pr[\,E(K,M) = C\,] \quad = \quad \Pr[\,E(K,M') = C\,]$$

where probabilities are over choice of K

In words:
each message is equally likely to map to a given ciphertext

In other words:
seeing a ciphertext leaks nothing about what message was encrypted

Does a substitution cipher meet this definition? No!

Shannon's security notion

Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

$$\Pr[\, E(K,M) = C \,] \;=\; \Pr[\, E(K,M') = C \,]$$

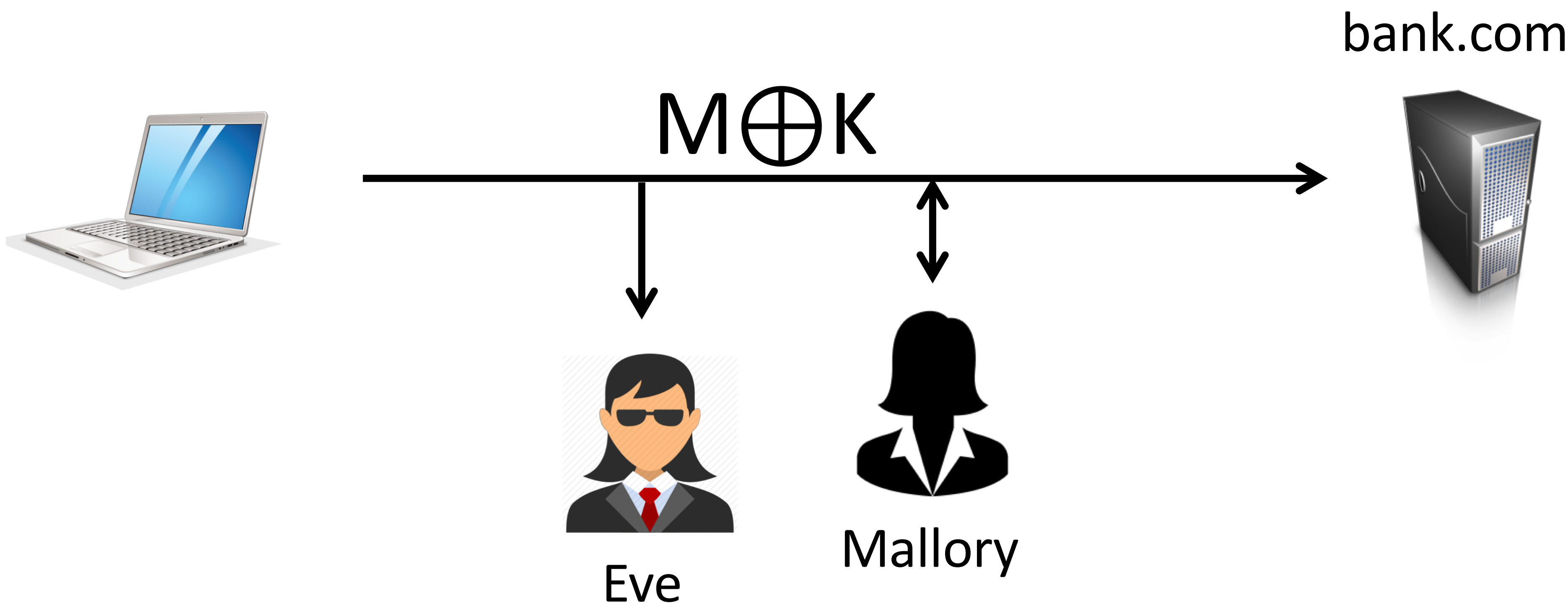
where probabilities are over choice of K

Thm. OTP is perfectly secure

For any C and M of length L bits

$$\Pr[\, K \oplus M = C \,] \;=\; 1 / 2^L$$

$$\Pr[\, K \oplus M = C \,] \;=\; \Pr[\, K \oplus M' = C \,]$$



K must be as large as M
Reusing K for M,M' leaks $M \oplus M'$
Message length is obvious
Mallory can make undetected modifications

OTP limitations

provable security

- Cryptography as a *computational science*
- Use computational intractability as basis for confidence

1. Design a cryptographic scheme
2. Provide a **proof** that no attacker with bounded computational resources can break it

[Goldwasser, Micali, Blum, 1980s]

Formal definitions

- Scheme semantics and assumption
- Security

Security Proofs (reductions)

Breaking scheme

$\Downarrow$

Breaking assumptions

provable security

- Provable security yields
  - well-defined assumptions and security goals
  - designers (and attackers) can focus on assumptions
- As long as assumptions hold, we can be confident in security of a cryptographic scheme

Typical assumptions

- Basic atomic primitives are hard to break:
  - Factoring of large composites intractable
  - RSA permutation hard-to-invert
  - Block ciphers (AES, DES) are good pseudorandom permutations (PRPs)
  - Hash functions are collision resistant

Confidence in atomic primitives is gained by cryptanalysis, public design competitions

SHA-3 competition, AES competition

recap

- Symmetric vs asymmetric cryptography
- Primitives
  - symmetric/asymmetric encryption
  - message authentication codes
  - digital signatures
  - key exchange
- Provable security
- Shannon's one-time pad
  - security guarantees and limitations

Cryptography as computational science

Use computational intractability as basis for confidence in systems

1. Design a cryptographic scheme
2. Provide proof that no attacker with limited computational resources can break it

Goldwasser, Micali and Blum circa 1980's

Formal definitions

Scheme semantics

Security

Security proofs (reductions)

Breaking scheme

$\Downarrow$

Breaking assumptions

Example:

Attacker can not recover credit card

$\Updownarrow$

Can not factor large composite numbers

But no one knows how to do this. It's been studied for a very long time!

As long as assumptions holds we believe in security of scheme!

Provable security yields

1) well-defined assumptions and security goals

2) attackers (cryptanalysts) can focus on assumptions